Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	16	contact adj management adj database	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:56
L2	41	(hash same (database adj record)) and synchroniz$3	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:56
L3	41	(hash same (database adj record)) and synchroniz$4	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:05
L4	26	3 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:03
L5	9	(("6678715") or ("6449640") or ("6321236") or ("5799072") or ("5790974") or ("5659741") or ("5649195") or ("5530861") or ("5204958")).PN.	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	OFF	2005/08/18 14:10
L6	1	5 and hash	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 14:16
L7	0	(compar$3 adj database adj record) and hash	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 14:17
L8	34	(compar$3 with (database adj record)) and hash	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 14:18
L9	13	8 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 14:19
L10	19	(modif$5 adj database adj record) and hash	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 14:19
L11	1	10 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 14:32
L12	16069	(chang$3 5 adj database adj record) and hash	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 14:19
L13	5	(chang$3 adj database adj record) and hash	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 14:31

L14	37	((unchanged unmodified) adj record) and synchroniz$5	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 14:32
L15	20	14 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 14:37
L16	24	(signature with (database adj record)) and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:11
L17	7	hash$3 with (chang$3 adj record)	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:02
L18	917	hash$3 with identical	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:02
L19	518	18 and database	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:03
L20	136	19 and synchroniz$	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:03
L21	39	20 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:03
L22	41	(hash$3 same (database adj record)) and synchroniz$4	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:10
L23	20	(first adj record) with (second adj record) with identical	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:11
L24	10	23 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:17
L25	183	(compar$3 adj hash$3)and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:30
L26	93	(content adj hash$3)and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:30
L27	60	26 and database	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:31
L28	71	26 and record	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:31

S1	10	("20020156798"|"5204958"|"5530861"|"5649195"|"5659741"|"5790974"|"5799072"|"6321236"|"6449640"|"6678715").PN.	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 11:40
S2	1	S1 and hash	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 11:42
S3	1	"6223187".pn.	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 11:52
S4	39	"6044381"	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/18 11:52
S5	111	("4432057" | "4807182" | "4819156" | "4827423" | "4866611" | "4875159" | "4956809" | "4980844" | "5065360" | "5136707" | "5142619" | "5155850" | "5170480" | "5187787" | "5210868" | "5228116" | "5237678" | "5251151" | "5251291" | "5261045" | "5261094" | "5272628" | "5278978" | "5278982" | "5283887" | "5293627" | "5301313" | "5315709" | "5327555" | "5333252" | "5333265" | "5333316" | "5339392" | "5339434" | "5355476" | "5375234" | "5392390" | "5396612" | "5434994" | "5444851" | "5463735" | "5475833" | "5511188" | "5519606" | "5560005" | "5568402" | "5583793" | "5600834" | "5613113" | "5615364" | "5619689" | "5630081" | "5666530" | "5666553" | "5682524" | "5684984" | "5684990" | "5701423" | "5708812" | "5708840" | "5710922" | "5727202" | "5729735" | "5745712" | "5758150" | "5758355" | "5778388" | "5790789" | "5845293" | "5870759" | "5870765" | "5884323" | "5884324" | "5884325" | "5897640" | "5926824").PN. OR ("6044381").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 11:57

S6	76	("4432057" | "4807182" | "4819156" | "4827423" | "4866611" | "4875159" | "4956809" | "4980844" | "5065360" | "5136707" | "5142619" | "5155850" | "5170480" | "5187787" | "5210868" | "5228116" | "5237678" | "5251151" | "5251291" | "5261045" | "5261094" | "5272628" | "5278978" | "5278982" | "5283887" | "5293627" | "5301313" | "5315709" | "5327555" | "5333252" | "5333265" | "5333316" | "5339392" | "5339434" | "5355476" | "5375234" | "5392390" | "5396612" | "5434994" | "5444851" | "5463735" | "5475833" | "5511188" | "5519606" | "5560005" | "5568402" | "5583793" | "5600834" | "5613113" | "5615364" | "5619689" | "5630081" | "5666530" | "5666553" | "5682524" | "5684984" | "5684990" | "5701423" | "5708812" | "5708840" | "5710922" | "5727202" | "5729735" | "5745712" | "5758150" | "5758355" | "5778388" | "5790789" | "5845293" | "5870759" | "5870765" | "5884323" | "5884324" | "5884325" | "5897640" | "5926824").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 12:45
S7	4	S6 and hash	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 12:46
S8	7	hash with (chang$3 adj record)	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:02
S9	7	hash with (modif$4 adj record)	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 12:48
S10	345	(database adj record) and synchroniz$3 and hash	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 15:02
S11	105	S10 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 14:17
S12	631	(compar$3 with hash) and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:56
S13	174	(compar$3 near hash) and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:36
S14	72	"5758355"	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:21

S15	1	"5758355".pn.	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:21
S16	7	S14 and hash	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:22
S17	21	"5809494"	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:32
S18	233	palm.as.	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:32
S19	73	S18 and synchroniz$5	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:32
S20	8	S19 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:32
S21	76	("5001628" | "5392390" | "5463772" | "5537592" | "5544356" | "5574859" | "5592669").PN. OR ("5727202").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:36
S22	23	S21 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:39
S23	41	unique adj record adj ID	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:39
S24	13	S23 and (@rlad<="19970911" @ad<="19970911")	US-PGPUB; USPAT; USOCR	OR	ON	2005/08/18 13:39

Terms used **hash synchroniz** Found **1,526** of **160,172**

Sort results by relevance
Display results expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Split-ordered lists: lock-free extensible hash tables
Ori Shalev, Nir Shavit
July 2003 **Proceedings of the twenty-second annual symposium on Principles of distributed computing**
Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, citings, index terms

We present the first lock-free implementation of an extensible hash table running on current architectures. It provides concurrent insert, delete, and search operations with an expected $O(1)$ cost. It consists of very simple code, easily implementable using only load, store, and compare-and-swap operations. The new mathematical structure at the core of our algorithm is *recursive split-ordering*, a way of ordering elements in a linked list so that they can be repeatedly "split" using ...

Keywords: Compare-and-Swap, Concurrent Data Structures, Hash Table, Non-blocking Synchronization, Real-Time

2 Session 3: High performance dynamic lock-free hash tables and list-based sets
Maged M. Michael
August 2002 **Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures**
Full text available: pdf(238.11 KB) Additional Information: full citation, abstract, references, citings, index terms

Lock-free (non-blocking) shared data structures promise more robust performance and reliability than conventional lock-based implementations. However, all prior lock-free algorithms for sets and hash tables suffer from serious drawbacks that prevent or limit their use in practice. These drawbacks include size inflexibility, dependence on atomic primitives not supported on any current processor architecture, and dependence on highly-inefficient or blocking memory management techniques.Building on ...

3 Removing unnecessary synchronization in Java
Jeff Bogda, Urs Hölzle
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, citings, index

terms

Java programs perform many synchronization operations on data structures. Some of these synchronization are unnecessary; in particular, if an object is reachable only by a single thread, concurrent access is impossible and no synchronization is needed. We describe an interprocedural, flow- and context-insensitive dataflow analysis that finds such situations. A global optimizing transformation then eliminates synchronizations on these objects. For every program in our suite of ten Java bench ...

4 An efficient meta-lock for implementing ubiquitous synchronization

Ole Agesen, David Detlefs, Alex Garthwaite, Ross Knippel, Y. S. Ramakrishna, Derek White

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(2.00 MB) Additional Information: full citation, abstract, references, citings, index terms

Programs written in concurrent object-oriented languages, especially ones that employ thread-safe reusable class libraries, can execute synchronization operations (lock, notify, etc.) at an amazing rate. Unless implemented with utmost care, synchronization can become a performance bottleneck. Furthermore, in languages where every object may have its own monitor, per-object space overhead must be minimized. To address these concerns, we have developed a meta-lock to mediate access to synchro ...

Keywords: concurrent threads, object-oriented language implementation, synchronization

5 Thin locks: featherweight synchronization for Java

David F. Bacon, Ravi Konuru, Chet Murthy, Mauricio Serrano

May 1998 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation**, Volume 33 Issue 5

Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, citings, index terms

Language-supported synchronization is a source of serious performance problems in many Java programs. Even single-threaded applications may spend up to half their time performing useless synchronization due to the thread-safe nature of the Java libraries. We solve this performance problem with a new algorithm that allows lock and unlock operations to be performed with only a few machine instructions in the most common cases. Our locks only require a partial word per object, and were implemented ...

6 OOPSLA onward!: Finding bugs is easy

David Hovemeyer, William Pugh

December 2004 **ACM SIGPLAN Notices**, Volume 39 Issue 12

Full text available: pdf(506.92 KB) Additional Information: full citation, abstract, references

Many techniques have been developed over the years to automatically find bugs in software. Often, these techniques rely on formal methods and sophisticated program analysis. While these techniques are valuable, they can be difficult to apply, and they aren't always effective in finding real bugs.*Bug patterns* are code idioms that are often errors. We have implemented automatic detectors for a variety of bug patterns found in Java programs. In this paper, we describe how we have used bug pa ...

7 Disk-tape joins: synchronizing disk and tape access

Jussi Myllymaki, Miron Livny

May 1995 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and**

modeling of computer systems, Volume 23 Issue 1

Full text available: pdf(1.23 MB) Additional Information: full citation, abstract, references, citings, index terms

Today large amounts of data are stored on tertiary storage media such as magnetic tapes and optical disks. DBMSs typically operate only on magnetic disks since they know how to maneuver disks and how to optimize accesses on them. Tertiary devices present a problem for DBMSs since these devices have dismountable media and have very different operational characteristics compared to magnetic disks. For instance, most tape drives offer very high capacity at low cost but are accessed sequentially, in ...

Keywords: concurrent I/O, join methods, tertiary storage

8 Speculative synchronization: applying thread-level speculation to explicitly parallel applications

José F. Martínez, Josep Torrellas

October 2002 **Proceedings of the 10th international conference on Architectural support for programming languages and operating systems**, Volume 36 , 30 , 37 Issue 5 , 5 , 10

Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, citings

Barriers, locks, and flags are synchronizing operations widely used programmers and parallelizing compilers to produce race-free parallel programs. Often times, these operations are placed suboptimally, either because of conservative assumptions about the program, or merely for code simplicity.We propose *Speculative Synchronization,* which applies the philosophy behind Thread-Level Speculation (TLS) to explicitly parallel applications. Speculative threads execute past active barriers, busy ...

9 Optimistic replication

Yasushi Saito, Marc Shapiro

March 2005 **ACM Computing Surveys (CSUR)**, Volume 37 Issue 1

Full text available: pdf(656.72 KB) Additional Information: full citation, abstract, references, index terms

Data replication is a key technology in distributed systems that enables higher availability and performance. This article surveys optimistic replication algorithms. They allow replica contents to diverge in the short term to support concurrent work practices and tolerate failures in low-quality communication links. The importance of such techniques is increasing as collaboration through wide-area and mobile networks becomes popular.Optimistic replication deploys algorithms not seen in tradition ...

Keywords: Replication, disconnected operation, distributed systems, large scale systems, optimistic techniques

10 Wide-area architecture and protocols: Hierarchical substring caching for efficient content distribution to low-bandwidth clients

Utku Irmak, Torsten Suel

May 2005 **Proceedings of the 14th international conference on World Wide Web**

Full text available: pdf(221.09 KB) Additional Information: full citation, abstract, references, index terms

While overall bandwidth in the internet has grown rapidly over the last few years, and an increasing number of clients enjoy broadband connectivity, many others still access the internet over much slower dialup or wireless links. To address this issue, a number of techniques for optimized delivery of web and multimedia content over slow links have been proposed, including protocol optimizations, caching, compression, and multimedia transcoding, and several large ISPs have recently begun to widel ...

Keywords: HTTP, WWW, compression, web caching, web proxies

11 Language support for lightweight transactions

Tim Harris, Keir Fraser

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11

Full text available: pdf(224.15 KB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent programming is notoriously difficult. Current abstractions are intricate and make it hard to design computer systems that are reliable and scalable. We argue that these problems can be addressed by moving to a declarative style of concurrency control in which programmers directly indicate the safety properties that they require. In our scheme the programmer demarks sections of code which execute within lightweight software-based transactions that commit atomically and exactly once. Th ...

Keywords: concurrency, conditional critical regions, non-blocking systems, transactions

12 Data size optimizations for java programs

C. Scott Ananian, Martin Rinard

June 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems**, Volume 38 Issue 7

Full text available: pdf(349.36 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a set of techniques for reducing the memory consumption of object-oriented programs. These techniques include analysis algorithms and optimizations that use the results of these analyses to eliminate fields with constant values, reduce the sizes of fields based on the range of values that can appear in each field, and eliminate fields with common default values or usage patterns. We apply these optimizations both to fields declared by the programmer and to implicit fields in the runti ...

Keywords: bitwidth analysis, embedded systems, field externalization, field packing, size optimizations, static specialization

13 Alias annotations for program understanding

Jonathan Aldrich, Valentin Kostadinov, Craig Chambers

November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11

Full text available: pdf(336.14 KB) Additional Information: full citation, abstract, references, citings, index terms

One of the primary challenges in building and evolving large object-oriented systems is understanding aliasing between objects. Unexpected aliasing can lead to broken invariants, mistaken assumptions, security holes, and surprising side effects, all of which may lead to software defects and complicate software evolution.This paper presents AliasJava, a capability-based alias annotation system for Java that makes alias patterns explicit in the source code, enabling developers to reason more effec ...

Keywords: aliasing, aliasjava, encapsulation, java, ownership types, type inference, uniqueness

14 Continuous profiling: where have all the cycles gone?

Jennifer M. Anderson, Lance M. Berc, Jeffrey Dean, Sanjay Ghemawat, Monika R. Henzinger, Shun-Tak A. Leung, Richard L. Sites, Mark T. Vandevoorde, Carl A. Waldspurger, William E. Weihl

November 1997 **ACM Transactions on Computer Systems (TOCS)**, Volume 15 Issue 4

Full text available: pdf(259.35 KB) Additional Information: full citation, abstract, references, citings, index terms

This article describes the Digital Continuous Profiling Infrastructure, a sampling-based profiling system designed to run continuously on production systems. The system supports multiprocessors, works on unmodified executables, and collects profiles for entire systems, including user programs, shared libraries, and the operating system kernel. Samples are collected at a high rate (over 5200 samples/sec. per 333MHz processor), yet with low overhead (1–3% slowdown for most workloads). A ...

Keywords: performance understanding, performance-monitoring hardware, profiling, program analysis

15 Virtualizing Transactional Memory

Ravi Rajwar, Maurice Herlihy, Konrad Lai

June 2005 **Proceedings of the 32nd Annual International Symposium on Computer Architecture ISCA '05**

Full text available: pdf(199.77 KB) Additional Information: full citation, abstract

Writing concurrent programs is difficult because of the complexity of ensuring proper synchronization. Conventional lock-based synchronization suffers from wellknown limitations, so researchers have considered non-blocking transactions as an alternative. Recent hardware proposals have demonstrated how transactions can achieve high performance while not suffering limitations of lock-based mechanisms. However, current hardware proposals require programmers to be aware of platform-specific resource ...

16 Continuous profiling: where have all the cycles gone?

Jennifer M. Anderson, Lance M. Berc, Jeffrey Dean, Sanjay Ghemawat, Monika R. Henzinger, Shun-Tak A. Leung, Richard L. Sites, Mark T. Vandevoorde, Carl A. Waldspurger, William E. Weihl

October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5

Full text available: pdf(2.29 MB) Additional Information: full citation, references, citings, index terms

17 Secure wireless protocols: On the security of wireless network access with enhancements

Lein Harn, Wen-Jung Hsin

September 2003 **Proceedings of the 2003 ACM workshop on Wireless security**

Full text available: pdf(263.94 KB) Additional Information: full citation, abstract, references, index terms

The security of the current 3G wireless protocols addresses the problems faced by the 2G systems, in addition to fulfilling the higher 3G security requirements mandated from operating in IP networks as well as voice networks. However, the approach adopted by the two most popular 3G mobile system forerunners, UMTS and cdma2000, leaves many areas for improvement. In this paper, we improve the security of the 3G protocols in network access by providing strong periodically mutual authentication, str ...

Keywords: 3G mobile network security and authentication, security

18 Compactly encoding unstructured inputs with differential compression
Miklos Ajtai, Randal Burns, Ronald Fagin, Darrell D. E. Long, Larry Stockmeyer
May 2002 **Journal of the ACM (JACM)**, Volume 49 Issue 3

Full text available: pdf(348.32 KB) Additional Information: full citation, abstract, references, citings, index terms

The subject of this article is *differential compression*, the algorithmic task of finding common strings between versions of data and using them to encode one version compactly by describing it as a set of changes from its companion. A main goal of this work is to present new differencing algorithms that (i) operate at a fine granularity (the atomic unit of change), (ii) make no assumptions about the format or alignment of input data, and (iii) in practice use linear time, use constant spa ...

Keywords: Delta compression, differencing, differential compression

19 A study of locking objects with bimodal fields
Tamiya Onodera, Kiyokuni Kawachiya
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, citings, index terms

Object locking can be efficiently implemented by bimodal use of a field reserved in an object. The field is used as a lightweight lock in one mode, while it holds a reference to a heavyweight lock in the other mode. A bimodal locking algorithm recently proposed for Java achieves the highest performance in the absence of contention, and is still fast enough when contention occurs.However, mode transitions inherent in bimodal locking have not yet been fully considered. The algorith ...

20 Security: Ariadne:: a secure on-demand routing protocol for ad hoc networks
Yih-Chun Hu, Adrian Perrig, David B. Johnson
September 2002 **Proceedings of the 8th annual international conference on Mobile computing and networking**

Full text available: pdf(308.15 KB) Additional Information: full citation, abstract, references, citings, index terms

a secure on-demand routing protocol for ad hoc networks.

Keywords: ad hoc network routing, routing, security